CUSTOMER LIFETIME VALUE

MODELLING USING R

This project is prepared by Group 1 and is created exclusively as deliverable for OPIM-5603 Statistics in Business Analytics course

Instructor: Professor Cuong Do

UCONN SCHOOL OF BUSINESS

GROUP 1 TEAM MEMBERS

TARUN KUMAR – FULL TIME MBA, CLASS 2021

BHARATH TATA – FULL TIME MBA CLASS 2021
The Problem:

Customer lifetime value (CLV) is one of the key stats likely to be tracked as part of a customer experience program. CLV is a measurement of how valuable a customer is to your company with an unlimited time span as opposed to just the first purchase. This metric helps you understand a reasonable cost per acquisition.

CLV is the total worth to a business of a customer over the whole period of their relationship. It’s an important metric as it costs less to keep existing customers than it does to acquire new ones, so increasing the value of your existing customers is a great way to drive growth.

If the CLV of an average coffee shop customer is $1,000 and it costs more than $1,000 to acquire a new customer (advertising, marketing, offers, etc.) the coffee chain could be losing money unless it pares back its acquisition costs.

Knowing the CLV helps businesses develop strategies to acquire new customers and retain existing ones while maintaining profit margins.

CLV is distinct from the Net Promoter Score (NPS) that measures customer loyalty, and CSAT that measures customer satisfaction because it is tangibly linked to revenue rather than a somewhat intangible promise of loyalty and satisfaction.
How Do You Measure CLV?

If you’ve bought a $40 Christmas tree from the same grower for the last 10 years, your CLV has been worth $400 to them. But as you can imagine, in bigger companies CLV gets more complicated to calculate.

Some companies don’t attempt to measure CLV, citing the challenges of segregated teams, inadequate systems, and untargeted marketing.

When data from all areas of an organization is integrated however, it becomes easier to calculate CLV.

CLV can be measured in the following way:

1. Identify the touchpoints where the customer creates the value
2. Integrate records to create the customer journey
3. Measure revenue at each touchpoint
4. Add together over the lifetime of that customer
At its simplest, the formula for measuring CLV is:

**Customer revenue - the costs of acquiring and serving the customer = CLV**

Ultimately, you don’t need to get bogged down in complex calculations – you just need to be mindful of the value that a customer provides over their lifetime relationship with you. By understanding the customer experience and measuring feedback at all key touchpoints, you can start to understand the key drivers of CLV.

**Explanation for the R-code used for the CLV Modelling**

The Customer Life Time Value model in R has been prepared with help of defining user define function to calculate Recency, Frequency and Monetary (RFM) Values for the Ecommerce transactions and their percentages. Later, a logistic regression model is used to predict the CLV value on a e-commerce dataset that used the RFM Scoring

**Function#1**

New_Data_Frame (df, start_Date, end_Date, cust_ID , trans_Date , Amount_per_trans)
**Description**

The Function prepares the input data frame of Ecomm_df transaction records for RFM scoring.

a.) Remove duplicate records with same Customer ID.

b.) Find the most recent date per Customer ID, then calculate the days to the end_Date to get the Recency data.

c.) Calculate the quantity of transaction per customer to get the Frequency data.

d.) Sum up the amount spend by each customer and divide by Frequency to get the average amount spend per transaction, this becomes the Monetary data.

**Arguments explained**

df: A data frame of Ecomm_df transaction records with Customer ID, Dates, and the Amount of money spend per transaction.

start_Date: The start date of transaction; Records occur after the start date will be kept.

date: The end date of transaction; Records occur after the end date will be removed. It works with the start date to set a time.

Scope

cust_ID: column that contains Customer IDs in the input data frame.

trans_Date: column that contains the transaction dates in the input data frame.
Amount_per_trans: column that contains the amount spend by each customer per transaction in the input data frame.

**Function output**

A new data frame with 3 new columns: "Recency", "Frequency", and "Monetary".

Recency: The number of days from the most recent transaction of a customer to end_Date.

Frequency: The number of transactions of a customer during the period between start_Date and end_Date.

Monetary: The average amount of money spend per transaction by a customer during the period between start_Date and end_Date.
```r
# Function for data cleaning and re-arrangement

New_Data_Frame <- function(df, start_Date, end_Date, cust_ID = "ID", trans_Date = "Date", Amount_per_trans = "Amount") {

  df <- df[order(df[, trans_Date], decreasing = TRUE), ] # sort data frame by date descendingly

  # remove records outside date range [start_Date, end_Date]
  df <- df[df[, trans_Date] >= start_Date,]
  df <- df[df[, trans_Date] <= end_Date,]

  # remove rows with duplicate Customer ID
  newdf <- df[!duplicated(df[, cust_ID]),]

  # calculation for the Recency to the end_Date; Trying to get the most recent transaction with smaller value of recency
  Recency <- as.numeric(difftime(end_Date, newdf[, trans_Date], units = "days"))

  newdf <- cbind(newdf, Recency) # combining Recency column to the newdf data frame

  newdf <- newdf[order(newdf[, cust_ID]),] # sort data frame by ID

  # Frequency Calculation
  temp <- as.data.frame(table(df[, cust_ID]))
  Frequency <- temp[, 2]

  # Frequency Calculation
  temp <- as.data.frame(table(df[, cust_ID]))
  Frequency <- temp[, 2]

  newdf <- cbind(newdf, Frequency) # combining Frequency to the newdf data frame

  # Monetary Calculation
  tmp <- as.data.frame(tapply(df[, Amount_per_trans], df[, cust_ID], sum))
  Monetary <- temp[, 1] / Frequency

  newdf <- cbind(newdf, Monetary) # combining Monetary to the newdf data frame

  return(newdf)
}
...
```
**Function#2**

RFM_Score (df, r = 5, f = 5, m = 5)

**Description**

Scoring the Recency, Frequency, and Monetary in r, f, and m into specified number of bins (5) independently. This function calls scoring ()

Arguments explained

df: A data frame returned by New_Data_Frame ()
r: The highest point of Recency.
f: The highest point of Frequency.
m: The highest point of Monetary.

**Function output**

A new data frame with 4 new columns of "R_score", "F_score", "M_score", and "Total_score".
```r
# Scoring the Recency, Frequency, and Monetary

RFM_Score <- function(df, r = 5, f = 5, m = 5) {
  if (r <= 0 || f <= 0 || m <= 0) return
  # sort data for Recency with Recency (ascending); Frequency (descending); Monetary (descending) respectively
  df <- df[order(df$Recency, -df$Frequency, -df$Monetary),]
  R_score <- scoring(df, "Recency", r)
  df <- cbind(df, R_score) # combine R_score to the df data frame
  # sort data set for Frequency with Recency (descending) - Frequency (ascending) - Monetary (descending) respectively
  df <- df[order(-df$Recency, df$Frequency, -df$Monetary),]
  F_score <- scoring(df, "Frequency", f)
  df <- cbind(df, F_score) # combine F_score to the df data frame
  # sort data set for Monetary with Recency (descending) - Frequency (ascending) - Monetary (ascending) respectively
  df <- df[order(-df$Recency, -df$Frequency, df$Monetary),]
  M_score <- scoring(df, "Monetary", m)
  df <- cbind(df, M_score) # combine M_score to the df data frame

  # sort data frame by R_score, F_score, and M_score in descending order
  df <- df[order(-df$R_score, -df$F_score, -df$M_score),]

  Total_score <- c(100 * df$R_score + 10 * df$F_score + df$M_score) # Total Score Calculation
  # combine Total_score to the df data frame
  df <- cbind(df, Total_score)

  return(df)
}

...
Function#3

scoring (df, column, r = 5)

Description

This function invokes by RFM_Score()

Recency: Lower recency value = More recent = Higher R_Score

Frequency: Lower freq. value = Lower return visits = Higher F_Score

Monetary: Lower monetary value = Lower spent / visit / customer = Higher M_Score.

```r
# The function scoring() is used by RFM_Score.
scoring <- function(df, column, r = 5) {

    len <- dim(df)[1] # Determined number of measures of data frame

    # create the number of rows found in passed-in data frame
    score <- rep(0, times = len)

    # Determine the quantity of rows per bin
    nr <- round(len / r)
    if (nr > 0) {
        r_Start <- 0
        r_End <- 0

        # iteration through each bin
        for (i in 1:r) {
            # initialize start row number and end row number
            r_Start = r_End + 1

            # skip one "i" if the r_Start is already in the scope.
            if (r_Start > i * nr) next

            # i is at last bin
        }
    }
}
```
Function#4

Score_Breaks (df, r, f, m)

Description

Scoring the Recency, Frequency, and Monetary in r, f, and m into certain bin, where number of bins are calculated based on series of breaks specified by user.

Arguments

df: A data frame returned by New_Data_Frame ()

r: A vector of Recency breaks

f: A vector of Frequency breaks

m: A vector of Monetary breaks
Function output

A new data frame with 4 new columns of "R_score", "F_score", "M_score", and "Total_score".

```r
# Scoring with breaks specified by user
Score_Breaks <- function(df, r, f, m) {
  # scoring the Recency
  len = length(r)
  R_Score <- c(rep(1, length(df[,1])))
  df <- cbind(df, R_Score)
  for(i in 1:len){
    if(i == 1){
      p1=0
    }else{
      p1=r[i-1]
    }
    p2=r[i]
  }
  if(dim(df[p1<df$Recency & df$Recency<=p2,])[1]>0) df[p1<df$Recency & df$Recency<=p2,]$R_Score = len - i + 2
  
  # scoring the Frequency
  len = length(f)
  F_Score <- c(rep(1, length(df[,1])))
  df <- cbind(df, F_Score)
  for(i in 1:len){
    if(i == 1){
      p1=0
    }else{
      p1=f[i-1]
    }
  }
  if(dim(df[p1<df$Frequency & df$Frequency<=p2,])[1]>0) df[p1<df$Frequency & df$Frequency<=p2,]$F_Score = 1
  if(dim(df[f[len]<df$Frequency,])[1]>0) df[f[len]<df$Frequency,$F_Score = len+1
  
  # scoring the Monetary
  len = length(m)
  M_Score <- c(rep(1, length(df[,1])))
  df <- cbind(df, M_Score)
  for(i in 1:len){
    if(i == 1){
      p1=0
    }else{
      p1=m[i-1]
    }
    p2=m[i]
  }
  if(dim(df[p1<df$Monetary & df$Monetary<=p2,])[1]>0) df[p1<df$Monetary & df$Monetary<=p2,]$M_Score = 1
  if(dim(df[m[len]<df$Monetary,])[1]>0) df[m[len]<df$Monetary,$M_Score = len+1
  
  # Sorting the dataframe by R_Score, F_Score, and M_Score in descending order
  df <- df[order(-df$R_Score, -df$F_Score, -df$M_Score),]
  
  # Total score calculation
  Total_Score <- c(100*df$R_Score + 10*df$F_Score + df$M_Score)
  df <- cbind(df, Total_Score)

  return(df)
}
```
Function#5

Percentages (df, colNames)

Description

Calculate the probabilities of "Buy"/Repurchase grouped by R, F, M values respectively or in combination.

Arguments

df: A data frame returned by New_Data_Frame() with respect to calculation of Recency, Frequency, and Monetary along with its scores.

colNames: a vector of column names to be grouped by such as "Recency" or the combination of "Recency" and "Frequency"

Function Output

Data frame with the variables being used to grouped by and the percentages of customers who buy accordingly
Function#6

CLV (r, f, rev, cost, n, periods, dr, pModel)

Description

Calculate Customer Life Time Value based on Recency and Frequency

Arguments

r: Recency value (e.g., \( r = 0 \)).

f: Frequency value (e.g., \( f = 1 \)).

rev: Anticipated/Expected revenue from customer.

n: Num. of customers with the same Recency and Frequency value.
cost: Associated cost per period for each potential customer (Buy or No-Buy).

periods: Num. of period(s) customer will stay before churning.

dr: Discount Rate.

pModel: Regression model used to predict the "Buy" rate based on Recency, Frequency, or Monetary.

**Function output**

Customer's value after n periods

```r
# Calculation for Customer Life Time Value based on Recency and Frequency after n periods

CLV <- function(r, f, rev, cost, n, periods, dr, pModel) {
  df <- data.frame(period = c(0), r = c(r), f = c(f), n = c(n), value = c(0))

  for (i in 1:periods) {
    backstep <- df[df$period == i-1,]
    nrow <- nrow(backstep)

    for (j in 1:nrow) {
      r <- backstep[j,]$r
      f <- backstep[j,]$f
      n <- backstep[j,]$n
      p <- predict(pModel, data.frame(Recency = r, Frequency = f), type = "response")[1]
      buyers <- n * p

      # Predict "Buy" probability for this period
      df <- rbind( df, c(i, 0, f+1, buyers, buyers*(rev-cost) / (1+dr)^i ) )

      # Predict "No-Buy" probability for this period
      df <- rbind( df, c(i, r+1, f, n-buyers, (n-buyers)*(0-cost) / (1+dr)^i ) )
    }
  }
  return(sum(df$value))
}
```

...
Steps for the main program code

1) Importing the raw e-commerce transaction data

```r
# Import raw data with 69,659 customer transactions
getwd()

# Importing directly using import function in R
Ecomm_df <- Ecomm_trans_data
head(Ecomm_df)

# Data for the date range from 01/01/1997 to 06/30/1998
Ecomm_df <- as.data.frame(cbind(Ecomm_df[,1], Ecomm_df[,2], Ecomm_df[,4]))
names <- c("ID", "Date", "Amount")
names(Ecomm_df) <- names

# Transform Date column from text to date format
Ecomm_df[,2] <- as.Date( as.character(Ecomm_df[,2]), "%Y-%m%d")
head(Ecomm_df)
str(Ecomm_df)
```

Resulting Data frame

<table>
<thead>
<tr>
<th>ID &lt;dbl&gt;</th>
<th>Date &lt;date&gt;</th>
<th>Amount &lt;dbl&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1997-01-01</td>
<td>11.77</td>
</tr>
<tr>
<td>2</td>
<td>1997-01-12</td>
<td>12.00</td>
</tr>
<tr>
<td>3</td>
<td>1997-01-12</td>
<td>77.00</td>
</tr>
<tr>
<td>4</td>
<td>1997-01-02</td>
<td>20.76</td>
</tr>
<tr>
<td>5</td>
<td>1997-03-30</td>
<td>20.76</td>
</tr>
<tr>
<td>6</td>
<td>1997-04-02</td>
<td>19.54</td>
</tr>
</tbody>
</table>

6 rows

[1] "D:/UCONN FT MBA/MBA SEM 1 Subjects/UCONN Business Analytics OPIM 5603/Group001 R Project"
'data.frame': 69659 obs. of 3 variables:
$ ID : num 1 2 2 3 3 3 3 3 4 ...
$ Date : Date, format: "1997-01-01" "1997-01-12" ...
$ Amount: num 11.8 12 77 20.8 20.8 ...
2) Establishing Time-Frame for Customer Life Time Value Calculations

```r
# Establishing time frame for CLV calculation
# Historical transactional data with 18 months timeframe. This dataset will act as Training Data
startDate_Hist <- as.Date("19970101", "%Y%m%d")
endDate_Hist <- as.Date("19980228", "%Y%m%d")

# Forecast transactional data with 2 months time frame
startDate_Forecast <- as.Date("19980301", "%Y%m%d")
endDate_Forecast <- as.Date("19980430", "%Y%m%d")

# Historical Data set with distinct customers
Hist_df <- New_Data_Frame(Ecomm_df, startDate_Hist, endDate_Hist)

head(Hist_df)
str(Hist_df) # found 23,570 distinct customers with 18 months time-frame

# Dataset for Forecast with distinct customers within 2 months time-frame
Forecast_df <- New_Data_Frame(Ecomm_df, startDate_Forecast, endDate_Forecast)
head(Forecast_df)
str(Forecast_df) # found 2,979 distinct customers with 2 months time-frame
```

### Resulting Data frames

**Training data set**

<table>
<thead>
<tr>
<th>ID</th>
<th>Date</th>
<th>Amount</th>
<th>Recency</th>
<th>Frequency</th>
<th>Monetary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1997-01-01</td>
<td>11.77</td>
<td>423</td>
<td>1</td>
<td>11.77000</td>
</tr>
<tr>
<td>2</td>
<td>1997-01-12</td>
<td>12.00</td>
<td>412</td>
<td>2</td>
<td>44.50000</td>
</tr>
<tr>
<td>8</td>
<td>1997-11-25</td>
<td>20.96</td>
<td>95</td>
<td>5</td>
<td>27.89400</td>
</tr>
<tr>
<td>13</td>
<td>1997-12-12</td>
<td>26.48</td>
<td>78</td>
<td>4</td>
<td>25.12500</td>
</tr>
<tr>
<td>24</td>
<td>1998-01-03</td>
<td>37.47</td>
<td>56</td>
<td>11</td>
<td>35.05545</td>
</tr>
<tr>
<td>25</td>
<td>1997-01-01</td>
<td>20.99</td>
<td>423</td>
<td>1</td>
<td>20.99000</td>
</tr>
</tbody>
</table>

6 rows

**Structure for Training data set**

'data.frame': 23570 obs. of 6 variables:
$ ID : num 1 2 3 4 5 6 7 8 9 10 ... 
$ Date : Date, format: "1997-01-01" "1997-01-12" ... 
$ Amount: num 11.8 12 21 26.5 37.5 ... 
$ Recency: num 423 412 95 78 56 423 140 65 291 403 ... 
$ Frequency: num 1 2 5 4 11 1 2 7 2 1 ... 
$ Monetary: num 11.8 44.5 27.9 25.1 35.1 ...
Forecast Data Set

<table>
<thead>
<tr>
<th>ID</th>
<th>Date</th>
<th>Amount</th>
<th>Recency</th>
<th>Frequency</th>
<th>Monetary</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>1998-03-22</td>
<td>138.50</td>
<td>39</td>
<td>1</td>
<td>138.500</td>
</tr>
<tr>
<td>36</td>
<td>1998-03-29</td>
<td>24.46</td>
<td>32</td>
<td>1</td>
<td>24.460</td>
</tr>
<tr>
<td>69</td>
<td>1998-04-17</td>
<td>12.99</td>
<td>13</td>
<td>1</td>
<td>12.990</td>
</tr>
<tr>
<td>91</td>
<td>1998-04-26</td>
<td>48.45</td>
<td>4</td>
<td>1</td>
<td>48.450</td>
</tr>
<tr>
<td>96</td>
<td>1998-04-09</td>
<td>38.06</td>
<td>21</td>
<td>2</td>
<td>40.475</td>
</tr>
<tr>
<td>100</td>
<td>1998-03-11</td>
<td>11.88</td>
<td>50</td>
<td>1</td>
<td>11.880</td>
</tr>
</tbody>
</table>

6 rows

Structure for forecast data set

```
'data.frame': 2979 obs. of 6 variables:
$ ID    : num 7 8 25 29 31 32 33 39 40 47 ...
$ Date  : Date, format: "1998-03-22" "1998-03-29" ...
$ Amount: num 138.5 24.5 13 48.5 38.1 ...
$ Recency: num 39 32 13 4 21 50 48 50 53 16 ...
$ Frequency: int 1 1 1 1 2 1 2 2 1 3 ...
$ Monetary: num 138.5 24.5 13 48.5 40.5 ...
```

3) Preparing transactional data records for CLV Calculations

```r

# Prepare the transaction data (Ecomm_df) records for CLV calculation.

# Setting purchasing cycle to specified forecast period
Forecast_Period <- as.numeric(difftime(endDate_Forecast, startDate_Forecast))
Hist_df$Recency <- Hist_df$Recency %/% Forecast_Period

# Categorize Monetary values into bins with size of $10 each
breaks <- seq(from = 0, to = 112, by = 10)
Hist_df$Monetary <- as.numeric(cut(Hist_df$Monetary, breaks, labels = FALSE))

# Adding "Buy" or "No Buy" column to the training/historical dataset
Buy <- rep(0, nrow(Hist_df))
Hist_df <- cbind(Hist_df, Buy)

# Identifying customers who purchased during the forecast period
Hist_df[Hist_df$ID %in% Forecast_df$ID, ]$Buy <- 1

# Create Training data set with specific conditions
Training_df <- Hist_df
head(Training_df)
...
Resulting Data Frames

<table>
<thead>
<tr>
<th>ID</th>
<th>Date</th>
<th>Amount</th>
<th>Recency</th>
<th>Frequency</th>
<th>Monetary</th>
<th>Buy</th>
<th>Buy</th>
<th>Buy</th>
<th>Buy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1997-01-01</td>
<td>11.77</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1997-01-12</td>
<td>12.00</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1997-11-25</td>
<td>20.96</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1997-12-12</td>
<td>26.48</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1998-01-03</td>
<td>37.47</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1997-01-01</td>
<td>20.99</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

6 rows | 1-10 of 12 columns

4) Calculating and Plotting RFM Percentages

```{r}
library(plyr)
# Calculations for the "Buy" percentage based on Recency
pRecency <- Percentages(Training_df, "Recency")

# Calculations for the "Buy" percentage based on Frequency
pFreq <- Percentages(Training_df, "Frequency")

# Calculations for the "Buy" percentage based on Monet
pMonetary <- Percentages(Training_df, "Monetary")

# plot and draw fit curves of Percentage
par( mfnrow = c(1, 3), oma = c(0,0,2,0) )
plot(pRecency$Recency, pRecency$Percentage * 100, xlab = "Recency", ylab = "Prob of Purchasing (%)")
lines(lowess(pRecency$Recency, pRecency$Percentage * 100), col="blue", lty = 2)
plot(pFreq$Frequency, pFreq$Percentage * 100, xlab = "Frequency", ylab = "Prob of Purchasing (%)")
lines(lowess(pFreq$Frequency, pFreq$Percentage * 100), col="blue", lty = 2)
plot(pMonetary$Monetary, pMonetary$Percentage * 100, xlab = "Monetary", ylab = "Prob of Purchasing (%)")
title("Percentages ~ (Recency, Frequency, Monetary)", y=10, outer=TRUE)

# logistics regression on Purchase Pctg ~ Recency
r.glm = glm(Percentage~Recency, family = quasibinomial(link = "logit"), data = pRecency)

# logistics regression on Purchase Pctg ~ Frequency
f.glm = glm(Percentage~Frequency, family = quasibinomial(link = "logit"), data = pFreq)
```
# logistics regression on Purchase Pctg ~ Frequency
f glm = glm(Percentage~Frequency, family = quasibinomial(link = "logit"), data = pFreq)

# logistics regression on Purchase Pctg ~ Monetary
m glm = glm(Percentage~Monetary, family = quasibinomial(link = "logit"), data = pMonetary)

par( mfrow = c(1, 1) )

model <- glm(Buy ~ Recency + Frequency, data = Training_df, family = quasibinomial(link = "logit"))
pred_01 <- predict(model, data.frame(Recency = c(0), Frequency = c(1), type = "response")
pred_01
pred_02 <- predict(model, data.frame(Recency = c(0), Frequency = c(2), type = "response")
pred_02
pred_03 <- predict(model, data.frame(Recency = c(0), Frequency = c(3), type = "response")
pred_03

Resulting Plots

![Resulting Plots](image-url)
4) Running Logistic Regression Model on Customer Purchase (Based on Recency, Frequency and Monetary Values)

```
Call: glm(formula = Buy ~ Recency + Frequency, family = quasibinomial(link = "logit"), 
         data = Training_df)

Coefficients:
(Intercept)    Recency    Frequency
       -1.2052       -0.3772       0.1485

Degrees of Freedom: 23569 Total (i.e. Null); 23567 Residual
Null Deviance: 17890
Residual Deviance: 13850    AIC: NA
```

5) Prediction of CLV Value Using Logistic Regression

```
```
# 5.) Calculate customer's value within the forecast period

# Set of observations
r = 0       # initial Recency state (e.g., 0)
f = 1       # initial Frequency state (e.g., 1)
Rev = 100   # Anticipated revenue from customer.
Cost = 0    # cost for each potential customer (Buy or No-Buy) per period.
n = 1       # Num. of customers with the same Recency and Frequency value.
periods = 3 # Num. of period(s) customer will stay before churning.
dr = 0.02   # Discount rate
model = LogReg_Results.Buy_RF

Cust_Value <- CLV(r, f, Rev, Cost, n, periods, dr, model)
Cust_Value
```

**Resulting Value**

```
[1] 19.33779
```